37/45

CS 271 - Project 0001

James Le

February 1, 2017

1. Consider the searching problem:

   Input: a sequence $A =< a_1, a_2, \ldots, a_n >$ and a value v

   Output: an index i such that $v = A[i]$ or NIL is v is not found

   Linear search algorithm scans through the sequence one element at a time until an element $A[i] = v$ is found, at which point it stops and returns i.

   (a) Write pseudocode for a linear search.

```
LinearSearch(A,v)
v = A[0]
for i = 1 to A.length
     if A[i] = v
         reutrn i
return NIL
```

   (b) Write a loop invariant for the loop in your algorithm.

   Before iteration i of the **for** loop, all values in the subarray A[0...i - 1] $\neq v$.

   (c) Prove that the loop invariant is correct.

   To prove that the loop invariant is correct, I need to show that the **Initialization** and **Maintenance** properties hold.

   Initialization Before the first loop iteration, the invariant holds since the statement is empty.

   -8

   Prove this by induction.

   Maintenance The loop invariant is maintained at each iteration. Otherwise, at the i-th iteration, there is a value $k < i$ such that $A[k] = v$. However, in that case, for the k-th iteration of the loop, the value k is returned and there is no i-th iteration of the loop. That is a contradiction.

   (d) Use the termination condition of your loop invariant to prove that your algorithm is correct.

   Termination When the loop terminates, there are two possible cases:

   1 - The loop terminates after $i \leq A.length$ iterations and returns i, in which case the **if** condition ensures that $A[i] = v$. & the L.I says that no earlier A[i] = v.

   2 - The loop continues when i exceeds A.length: By the loop invariant, for all $k \leq A.length$ and $A[k] \neq v$, the returning NIL is correct.

   (e) What are the best case, worst case, and average case time complexities for linear search, using θ notation? For the average case, assume v is equally likely to match any element in the array. Justify your answers.

   Let $T(n)$ be the running time of linear search algorithm. Thus

$$T(n) = c_1 n + c_2(n-1) + c_3 + c_4 = (c_1 + c_2)n + (c_3 + c_4 - c_2) = \theta(n)$$

   Best Case The first number in the sequence is v, then $T(n) = \theta(1)$.

   Worst Case v is the last number in the sequence or it does not exist at all, then I have to search through the whole list. Then $T(n) = \theta(n)$.

   -3

   Average Case The average case running time is also $\theta(n)$. why?

2. Consider the bubble sort algorithm:

```
BubbleSort(A,n)
for i = 1 to n - 1
for j = n downto i + 1
     if A[j] < A[j - 1] then
         exchange A[j] and A[j - 1]
```

(a) State precisely a loop invariant for the inner for loop, and prove that this loop invariant is correct.

Loop Invariant Before iteration j of the inner for loop, the smallest element in subarray $A[i...n]$ is in the subarray $A[i...j]$.

Initialization Before the first loop iteration when $j = n$, the invariant holds because the smallest element in subarray $A[i...n]$ is in the subarray $A[i...n]$.

Maintenance We need to show that the invariant holds for prior to iteration $j - 1$ as j goes downward.

If the smallest element of $A[i...n]$ was in $A[i...j-1]$ prior to iteration j, we are done because the smallest element never moves to the right. Therefore, prior to iteration $j-1$, the smallest element of $A[i...n]$ is in $A[i...j-1]$.

If the smallest element of $A[i...n]$ was not in $A[i...j-1]$ prior to iteration j, this means that it was in $A[j]$ prior to iteration j, because it has to be in $A[i..j]$ prior to iteration j. In that case, during iteration j, $A[j]$, being the smallest element, will be exchanged with $A[j-1]$, and from there on, that smallest element can never move to the right.

Therefore, prior to iteration $j - 1$, the smallest element in $A[i...n]$ is in $A[i...j-1]$.

Termination The loop ends when j exceeds $i + 1$ from below (when $j = i$). We have the smallest element of $A[i...n]$ is in $A[i]$.

(b) State a loop invariant for the outer for loop, and use the termination condition of the inner loop invariant to prove that the outer loop invariant is correct.

Loop Invariant Before iteration i of the outer for loop, the subarray $A[1...i]$ is sorted and any element in $A[i+1...n]$ is greater or equal to any element in $A[...i]$.

Initialization Before the first loop iteration, the invariant holds because the subarray $A[1...0]$ is sorted (no element).

Maintenance Given the subarray $A[1...k-1]$ is sorted. Iteration k inserts at position k the smallest of the remaining unsorted elements of $A[k...n]$, as computed by the j loop. $A[1..k-1]$ contains only elements smaller than $A[k..n]$, and $A[k]$ is smaller than any element in $A[k+1..n]$, then $A[1...k]$ is sorted and the invariant is maintained.

Termination At the last iteration, $A[1..n-1]$ is sorted, and all element in $A[n-1..n]$ are larger than elements in $A[1..n-1]$. Hence $A[1..n]$ is sorted.

(c) Use the termination condition of the outer loop invariant to prove that the algorithm is correct.

The loop ends i exceeds n, i.e. when $i = n + 1$. Then we have that the subarrray $A[i...n]$ contains the smallest elements of A in sorted order. As this is the whole array, the algorithm is correct.

(d) What are the best case and worst case time complexities for BubbleSort, in θ notation? Justify your answers.

Best Case The list is already sorted $\theta(n)$

Worst Case The inner loop is executed $(n-1)+(n-2)+(n-3)+...+0 = \frac{1}{2}n(n-1) = \theta(n^2)$

3. Use mathematical induction to prove that, for all integers $n \geq 0$,

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

Base Case: The base case is when $n = 0$. We have

$$\sum_{i=0}^{0} 2^i = 2 - 1 \to 2^0 = 1$$

which is true.

Inductive Step: Now let $k \geq 0$ be an arbitrary positive integer and assume that the statement is true for $n = k$. It means that

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1 .$$

~~As desired~~, by rewriting the sum we have

$$\sum_{i=0}^{k+1} 2^i = \sum_{i=0}^{k} 2^i + 2^{k+1} \overset{\text{by I.H.}}{=} 2^{k+1} - 1 + 2^{k+1} = 2^{(k+1)+1} - 1$$

Thus the statement holds for $n = k+1$. Hence, the statement holds for all $n \geq 0$ by induction.

4. Use mathematical induction to prove that, for all integers $n \geq 1$,

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Base Case: The base case is when $n = 1$. We have

$$1^2 = 1 = \frac{1.2.3}{6} = \frac{1(1+1)(2+1)}{6}$$

so the statement holds when $n = 1$.

Inductive Step: Now let $k \geq 1$ be an arbitrary positive integer and assume that the statement is true for $n = k$. It means that

$$\sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}$$

As desired, by rewriting the sum we have

$$\sum_{i=1}^{k+1} i^2 = \sum_{i=1}^{k} i^2 + (k+1)^2$$

$$= \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$

$$= \frac{(k+1)(k(2k+1) + 6(k+1))}{6}$$

$$= \frac{(k+1)(2k^2 + k + 6k + 6)}{6}$$

$$= \frac{(k+1)(2k^2 + 7k + 6)}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

$$= \frac{(k+1)((k+1)+1)(2(k+1)+1)}{6}$$

Thus the statement holds for $n = k+1$. Hence, the statement holds for all $n \geq 1$ by induction.